

T-count optimization and Reed-Muller codes

Matthew Amy*

*Institute for Quantum Computing, University of Waterloo, Waterloo, ON, Canada and
David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo, ON, Canada*

Michele Mosca†

*Institute for Quantum Computing, University of Waterloo, Waterloo, ON, Canada
Department of Combinatorics & Optimization,
University of Waterloo, Waterloo, ON, Canada
Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada and
Canadian Institute for Advanced Research, Toronto, ON, Canada*

We prove an equivalence between exactly minimizing the number of T gates in n -qubit quantum circuits over CNOT and T gates, together with the Clifford group powers of T , and minimum distance decoding in the length $2^n - 1$ punctured Reed-Muller code of order $n - 4$. The equivalence arises from an analysis of the particular form of phase polynomials generated by CNOT and T gates. As a consequence, we derive an algorithm for the optimization of T -count in $\{\text{CNOT}, T\}$ quantum circuits based on Reed-Muller decoders, along with a new upper bound of $O(n^2)$ on the number of T gates required to implement an n -qubit unitary over CNOT and T gates. We further generalize this result to show that minimizing finer angle Z -rotations corresponds to decoding lower order binary Reed-Muller codes. In particular, we show that minimizing the number of $R_z(\pi/2^k)$ gates in circuits over $\{\text{CNOT}, R_z(\pi/2^k)\}$ corresponds to minimum distance decoding in the length $2^n - 1$ punctured Reed-Muller code of order $(n - k - 2)$.

I. INTRODUCTION

The synthesis and optimization of quantum circuits has generated a great deal of interest in recent years. As qubit technologies become more stable and experimentalists increase the size of their systems, actually running algorithms on these machines becomes a practical concern. Moreover, we want to know how to *efficiently* run these algorithms on the given systems, or conversely how big and stable of a system we need to run a particular algorithm. Given the prevalence of the circuit model within quantum computing, quantum circuit optimization is an important tool in answering these questions.

Due to the great affect of noise on quantum computations in the standard circuit model, much research has shifted its focus from optimizing physical circuits to logical ones with respect to a fault-tolerance schemes meant to mitigate the errors due to this noise. These schemes usually have striking differences from physical gates in terms of resource costs. In particular, most of the common schemes implement Clifford group gates *transversally* – that is, by performing one physical gate on each physical qubit or group of qubits. This allows the logical operation to be performed precisely and with time proportional to the physical gate time. The additional operation needed to make a universal gate set is then typically implemented probabilistically with state distillation and gate teleportation, a less accurate procedure which requires both additional time and space compared to a single physical gate. The two qubit controlled-NOT (CNOT) gate, as an element of the Clifford group, is thus a relatively cheap operation in this paradigm, compared to the $T = \text{diag}(1, e^{i\frac{\pi}{4}})$ gate which is commonly chosen as the canonical non-Clifford gate. This is a reversal of the computational costs inherent in most physical implementations, where entangling gates are typically more difficult to implement than single qubit rotations, and hence requires different circuit optimizations. While alternative fault-tolerance methods such as Paetznick and Reichardt’s completely transversal Clifford+ T

* meamy@uwaterloo.ca

† michele.mosca@uwaterloo.ca

scheme [2] and anyonic quantum computing [3] are gaining in popularity, minimization of the number of T gates – called the T -count – in quantum circuits is an important and widely studied goal.

We build on previous work by Amy, Maslov and Mosca on the reduction of T gates in quantum circuits. In [4] it was shown that unitaries implementable over CNOT and T gates may be described in the computational basis as a bit-wise (linear) permutation together with a phase rotation that is an 8th root of unity to a power given by a pseudo-Boolean function of the input bits. This function, called the circuit’s *phase polynomial*, was shown to be expressible as a sum of linear Boolean functions, the multiplicity of each function giving the power of the T gate required to produce the term. This idea was later used in [1] to optimize both T -count and T -depth – the number of stages of parallel T -gates in a circuit – by computing a circuit’s phase polynomial, simplifying it, then synthesizing a new circuit from the polynomial with maximally parallelized T gates. While their benchmarks showed significant reduction of T gates, it was noted that this approach was not optimal, as it was shown that there exist distinct phase polynomials that give rise to the same unitary. In particular, it was observed that for all $x_1, x_2, x_3, x_4 \in \mathbb{F}_2$,

$$e^{i\frac{\pi}{4} \sum_{f \in \mathbb{F}_2^4 \rightarrow \mathbb{F}_2} f(x_1, x_2, x_3, x_4)} = 1 = e^0,$$

where $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is the space of all n -bit linear Boolean functions. It was left as an open question as to whether there exist other such identities, and whether such identities can be used to further reduce instances of T gates.

In this paper we prove that the identities on n qubits that are useful for reducing T -count is exactly the length $2^n - 1$ punctured Reed-Muller code of order $n - 4$. In doing so, we derive a new T -count optimization algorithm based on Reed-Muller decoding which is optimal for CNOT and T gate circuits when a minimum distance decoder is used. We implemented this optimization algorithm as a module in the quantum circuit optimizer T -par [5] and tested it on general Clifford+ T circuits with different Reed-Muller decoders. The results show modest savings in T -count while still remaining tractable for circuits of non-trivial size. As a consequence of this equivalence, we also obtain a new upper bound on the number of T gates required to implement a circuit over $\{\text{CNOT}, T\}$, as well as evidence to the intractability of exact T -count minimization through a polynomial-time equivalence to the minimum distance decoding problem for the length $2^n - 1$ punctured Reed-Muller code of order $n - 4$.

Our proof naturally generalizes to the case when the T gate is replaced with a Z rotation by any angle of the form $\pi/2^k$. These gate sets are closely related to the Clifford-cyclotomic gate sets studied in [6], and are widely used in quantum algorithms including Shor’s algorithm [7]. We show that the number of $\pi/2^k$ rotation gates for each value of k can be minimized by decoding Reed-Muller codes of different orders, opening up the possibility of optimizing such circuits at the high level, before decomposing them into a lower level gate set such as Clifford+ T . Further generalization to rotation angles that are arbitrary roots of unity is also possible, with any composite order $2^x 3^y 5^z \dots$ reducing to the case for a root of order 2^x .

A. Related work

Much work has gone into T -count and depth reduction in recent years. Amy *et al.* [4] identified the T -count and T -depth as important quantities in the efficiency of a logical quantum circuit, and gave new implementations of 2–4 bit quantum operations reducing T -count and depth from the previously best known. Their search-based algorithm was later optimized and extended by Gosset *et al.* [8] to directly optimize T -depth, leading to proofs of T -depth minimality for various 2–4 bit circuits. Selinger [9] showed that the Toffoli gate, as well as a general class of Clifford+ T circuits, can be parallelized to T -depth 1 with sufficiently many ancillas. Constructions for adding controls to quantum gates were also given which lowered the T -count and depth compared to best known practices using Toffoli gates. Amy, Maslov and Mosca later used similar ideas to create an automated, polynomial-time tool for reducing and parallelizing T gates called T -par, which uses matroid partitioning to parallelize the T gates. More recently, Abdessaied, Soeken and Drechsler [10] studied the effect of Hadamard gates on T -count and depth reductions, developing a tool that reduces Hadamard gates in quantum circuits leading to further T gate optimizations. Maslov [11] examined Toffoli gate implementations up to relative phase and used them to develop new designs for multiple control Toffolis using fewer ancillas, CNOT, and in some cases T gates, than standard designs.

A great deal of work optimizing T -count and depth in single qubit circuits has also been done recently, with series of works on exact [12] and approximate [13–15] minimal synthesis, as well as repeat-until-success circuits [16, 17].

B. Overview

The rest of the paper is organized as follows. Section II gives definitions and notation that will be used throughout the paper. Section III defines the linear phase operators, details their representation as weighted sums of linear Boolean functions and synthesis. Section IV defines an additive subgroup of $\mathbb{Z}_8^{2^n-1}$ whose cosets correspond to the unique linear phase operators, then characterizes its binary residue as a Reed-Muller code and gives applications. Section V formally proves the equivalence of this (binary residue) subgroup with the length $2^n - 1$ punctured Reed-Muller code of order $n - 4$. Finally Section VI generalizes the result to circuits over CNOT gates and phase rotations with angles of the form $\pi/2^k$, and Section VII details the experimental evaluation of our technique.

II. PRELIMINARIES

We assume basic knowledge of quantum computing, but no knowledge of coding theory. For background on quantum computing, the reader is referred to [18].

A. Quantum gates

In this paper we will primarily be interested in two gates: the controlled-NOT ($\text{CNOT} : |x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle$) and the T -gate ($T : |x\rangle \mapsto e^{i\frac{\pi}{4}x}|x\rangle$). These two gates, together with $P := T^2$ and $Z := T^4$ gates, comprise what we refer to for brevity as the $\{\text{CNOT}, T\}$ gate set. We include the P and Z gates in this set to distinguish them from sequences of T gates which are generally much more expensive to implement in most fault-tolerance schemes. Given any power $k \in \mathbb{Z}_8$ of the T gate, we define a minimal T -gate expansion by

$$T^k := Z^{k_2} P^{k_1} T^{k_0}$$

where $k_2 k_1 k_0$ is the binary expansion of k . Note that $T^8 = I$ so $T^k = T^{k \bmod 8}$ for all integers k .

By a circuit over a particular set of gates we mean a sequential list of gates taken from the set, each with a list of qubits the gate is to be applied to. In this way, two distinct circuits may correspond to the same unitary matrix – we call such circuits *equivalent* in this case. The problem of optimizing quantum circuits is then to find, given a circuit, an equivalent circuit minimizing some cost function. In this work we consider optimizing strictly the T -count of quantum circuits, defined as the number of T gates appearing in a circuit. In particular, we define the problem of T -gate minimization as follows:

Problem 1 (T -gate minimization). *Given a circuit over a gate set containing the T gate, find an equivalent circuit over the same gate set with a minimal number of T gates.*

For completeness we also define the Hadamard gate in sum-over-paths style,

$$H : |x\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{x' \in \{0,1\}} (-1)^{x'} |x'\rangle.$$

The addition of the Hadamard gate to the $\{\text{CNOT}, T\}$ gate set gives a universal set, in the sense that any unitary operator may be approximated efficiently with gates in this set. These gates form a set of generators of the *Clifford+T* gate set commonly used in fault-tolerant quantum computing.

B. Coding theory

We provide only the necessary definitions from coding theory – for a more complete introduction, see MacWilliams & Sloane [19].

A length n *binary linear code* is a subspace C of \mathbb{F}_2^n , where \mathbb{F}_2 is the unique 2-element field $(\{0, 1\}, \oplus, \cdot)$ with addition (\oplus) and multiplication (\cdot) modulo 2. The vectors of C are called the *codewords* of C . Note that \mathbb{F}_2 is the set of Boolean values with addition corresponding to exclusive-OR and multiplication corresponding to AND. We also define Boolean negation in \mathbb{F}_2 as $\bar{x} := 1 \oplus x$. Addition, multiplication and negation are extended to vectors component-wise – that is, $\mathbf{x}\mathbf{y}$ is the component-wise multiple of vectors \mathbf{x} and \mathbf{y} , as opposed to vector space multiplication.

We denote binary vectors by boldface letters e.g., $\mathbf{x} = x_1x_2 \cdots x_n \in \mathbb{F}_2^n$, and use them interchangeably as bit strings. In particular, we denote the n -qubit computational basis vectors by $|\mathbf{x}\rangle$ where \mathbf{x} is a binary vector/bit string. The (*Hamming*) *weight* of a binary vector, denoted $\text{wt}(\mathbf{x})$, $\mathbf{x} \in \mathbb{F}_2^n$, is defined as the number of non-zero entries it contains, and the (*Hamming*) *distance* between two binary vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ is the weight of their sum:

$$d(\mathbf{x}, \mathbf{y}) := \text{wt}(\mathbf{x} \oplus \mathbf{y}).$$

Given a received vector $\mathbf{x} \oplus \mathbf{e}$ where $\mathbf{x} \in C$ and $\mathbf{e} \in \mathbb{F}_2^n$ is some error vector, we wish to find \mathbf{x} – this process is known as *decoding*. In this work, we are only concerned with *minimum distance decoding*, as it relates directly to T -count optimization.

Problem 2. (*Minimum distance decoding for a binary linear code C*) Given a received vector $\mathbf{x} \in \mathbb{F}_2^n$, find a codeword $\mathbf{y} \in C$ such that for all $\mathbf{z} \in C$, $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z})$.

This problem is closely related to the more general *closest vector problem* (CVP) over a lattice, and in fact coincides with the CVP problem over the lattice C with the Hamming weight as the norm. Minimum distance decoding is commonly studied as it reasonably approximates maximum likelihood decoding when bit flip errors are independent of one another.

We give one more definition from coding theory which will be relevant to our work: the maximum distance of any vector from a codeword, called the *covering radius*.

Definition 1. The *covering radius* of a length n binary code C is

$$\rho(C) = \max_{\mathbf{x} \in \mathbb{F}_2^n} \min_{\mathbf{y} \in C} d(\mathbf{x}, \mathbf{y}).$$

C. Reed-Muller codes

Many different presentations of the binary Reed-Muller codes ([20, 21]) are known; we use a presentation based on multivariate polynomials as it will provide a convenient setting for our proofs. For more details the reader is referred to [19].

Let $\mathbb{F}_2[v_1, v_2, \dots, v_n]$ be the ring of polynomials in n variables over \mathbb{F}_2 . We use the symbols v_1, v_2, \dots, v_n to denote Boolean variables so as to differentiate them from elements of binary vectors. Given $f \in \mathbb{F}_2[v_1, v_2, \dots, v_n]$ we define the *evaluation vector* of f , when viewed as an n -ary function, to be the length $2^n - 1$ vector of evaluations of f for inputs of non-zero Hamming weight – i.e.

$$(f(1, 0, \dots, 0), f(0, 1, \dots, 0), \dots, f(1, 1, \dots, 1)).$$

We denote the evaluation vector of a polynomial function f by \mathbf{f} . Since $v^2 = v$ for all $v \in \mathbb{F}_2$, without loss of generality we only consider polynomials $f \in \mathbb{F}_2[v_1, v_2, \dots, v_n]$ that contain binary exponents 0 or 1 on variables. Identifying the variable v_i with the Boolean function $f(v_1, v_2, \dots, v_n) = v_i$, we denote the evaluation vector of v_i by \mathbf{v}_i . It can be easily verified that for any Boolean polynomial $f = \bigoplus_{\mathbf{y} \in \mathbb{F}_2^n} v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n}$, the evaluation vector of f is equal to $\bigoplus_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ – again, exponentiation of a Boolean vector is defined as component-wise exponentiation.

	(1, 0, 0, ..., 0)	(0, 1, 0, ..., 0)	(1, 1, 0, ..., 0)	(0, 0, 1, ..., 0)	...	(1, 1, 1, ..., 1)
1	1	1	1	0	...	1
v_1	1	0	1	0	...	1
v_2	0	1	1	0	...	1
$v_1 v_2$	0	0	1	0	...	1
v_3	0	0	0	1	...	1
\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
$v_1 v_2 \cdots v_n$	0	0	0	0	...	1

TABLE I. Evaluation vectors for monomials over n Boolean variables.

We define the *total degree* of a monomial $v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n}$ to be the sum of its exponents:

$$\deg(v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n}) = \sum_{i=1}^n y_i = \text{wt}(\mathbf{y}).$$

The degree of a polynomial function $f \in \mathbb{F}_2[v_1, v_2, \dots, v_n]$, denoted $\deg(f)$, is defined as the maximum total degree of each monomial. Table I illustrates the evaluation vectors of the 2^n monomials on n variables. Note that the set of non-constant monomial evaluation vectors forms a basis for the space $\mathbb{F}_2^{2^n-1}$.

Definition 2. The *punctured Reed-Muller code* of order r and length $2^n - 1$, denoted $\mathcal{RM}(r, n)^*$, is the set of evaluation vectors for all polynomials $f \in \mathbb{F}_2[v_1, v_2, \dots, v_n]$ of degree at most r .

The non-punctured, length 2^n Reed-Muller code of order r is defined in a similar fashion, using evaluation vectors consisting of all 2^n distinct evaluations for a given polynomial function instead.

D. Statement of main result

We now state the main result.

Theorem 1. *The T -count minimization problem over n -qubit, $\{\text{CNOT}, T\}$ circuits is polynomial-time (in 2^n) equivalent to minimum distance decoding for $\mathcal{RM}(n-4, n)^*$.*

We prove Theorem 1 by showing that the number of T gates in a $\{\text{CNOT}, T\}$ circuit corresponds directly to the number of odd-weight elements of a tuple in $\mathbb{Z}_8^{2^n-1}$. Moreover, this tuple can be computed from a circuit or vice versa in polynomial-time. An equivalence relation is then defined between tuples that implement the same unitary, which we show is in direct correspondence with cosets of the $\mathcal{RM}(n-4, n)^*$ code. We then use this equivalence to utilize the wealth of research performed on Reed-Muller decoding for quantum circuit optimization.

III. LINEAR PHASE OPERATORS

In this section we introduce linear phase operators as the sub class of unitaries implementable by $\{\text{CNOT}, T\}$ which require T gates. We review their representation as pseudo-Boolean polynomial functions and define the canonical T -count for a particular polynomial. Finally we show that a minimal T -count implementation of a linear phase operator corresponds to a minimal weight vector of a vector space coset.

We define $\mathcal{P}_4(n)$ to be the set of diagonal $2^n \times 2^n$ unitaries implementable over $\{\text{CNOT}, T\}$ – we restrict our attention to this subset as any circuit over $\{\text{CNOT}, T\}$ may be decomposed into a diagonal unitary followed by a permutation implementable using only CNOT gates. Amy *et al.* [4, Lemma 2] showed that each such unitary $U \in \mathcal{P}_4(n)$ has the effect of applying a pseudo-Boolean function P to a computational basis state $|\mathbf{x}\rangle$, viewed as a vector $\mathbf{x} = x_1 x_2 \cdots x_n \in \mathbb{F}_2^n$, and kicking the result into the phase:

$$U : |\mathbf{x}\rangle \mapsto e^{i\frac{\pi}{4}P(\mathbf{x})}|\mathbf{x}\rangle.$$

Moreover, it was shown that the *phase polynomial* $P : \mathbb{F}_2^n \rightarrow \mathbb{Z}_8$ necessarily has a presentation as a weighted sum of (non-zero) linear Boolean functions:

$$P(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} \cdot \iota(y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n),$$

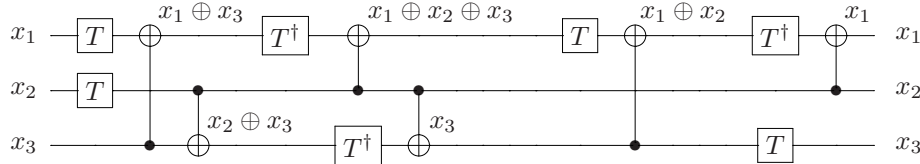
where the coefficients $a_{\mathbf{y}}$ are integers modulo 8. We call the $(2^n - 1)$ -tuple $\mathbf{a} = (a_1, a_2, \dots, a_{2^n - 1}) \in \mathbb{Z}_8^{2^n - 1}$ an *implementation* of P , and conversely denote the phase polynomial defined by a set of coefficients $\mathbf{a} \in \mathbb{Z}_8^{2^n - 1}$ by $P_{\mathbf{a}}$. As the function P involves both \mathbb{F}_2 and \mathbb{Z}_8 arithmetic, we use the natural inclusion ι of \mathbb{F}_2 in \mathbb{Z}_8 to coerce the binary valued result of $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$ into an integer – we leave the inclusion implicit whenever no confusion is likely to arise.

We call unitaries in $\mathcal{P}_4(n)$ $\pi/4$ *linear phase operators*, as they may be expressed as a sequence of $(\pi/4)$ phase rotations conditioned on linear Boolean functions of the input basis state. We drop the $\pi/4$ until Section VI when we generalize the result to $\pi/2^k$ linear phase operators. Given a particular phase polynomial P , we denote the linear phase operator with phase polynomial P by U_P .

Example 1. The doubly-controlled Z gate is a $\pi/4$ linear phase operator with phase function $P(x_1, x_2, x_3) = 4 \cdot x_1 x_2 x_3$. Using the identity $2 \cdot xy = x + y + 7(x \oplus y) \pmod{8}$ [9], the phase function may be presented as the following weighted sum of linear Boolean functions:

$$P(x_1, x_2, x_3) = x_1 + x_2 + 7(x_1 \oplus x_2) + x_3 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3).$$

Writing the coefficients above as a 7-tuple over \mathbb{Z}_8 we get $(1, 1, 7, 1, 7, 7, 1)$. Note that this implementation corresponds to the following circuit, taken from [1]. The state of each qubit after an update is shown to illustrate the relation between the state of a qubit as a Boolean function of the inputs and the application of phase gates.



Amy, Maslov and Mosca [1] showed that a linear phase operator U_P can be synthesized over $\{\text{CNOT}, T\}$ given an implementation $\mathbf{a} \in \mathbb{Z}_8^{2^n - 1}$ of P in time polynomial in the number of non-zero entries of \mathbf{a} – moreover, this number is linear in the size of the circuit. Their procedure applies each (non-trivial) phase shift $e^{i\frac{\pi}{4} a_{\mathbf{y}} \cdot \iota(y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n)}$ by first computing $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$, then applying $T^{a_{\mathbf{y}}}$ and uncomputing $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$. Recall that

$$T^k := Z^{k_2} P^{k_1} T^{k_0}$$

where $k_2 k_1 k_0$ is the binary expansion of k . Since each $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$ is a linear function of the basis state $x_1 x_2 \cdots x_n$, each value $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$ may be computed solely with CNOT gates, giving a total T -count equal to the number of odd elements of \mathbf{a} – we call this the T -count of an implementation. While in this work we are only concerned with the T -count of the synthesized circuit, T -depth can be minimized while keeping T -count the same by parallelizing this process through matroid partitioning [1].

The authors used this synthesis algorithm to optimize T -count in $\{\text{CNOT}, T\}$ circuits by first computing an implementation (set of coefficients) for the associated linear phase operator U_P from the circuit in polynomial time, then synthesizing an equivalent circuit. The remaining linear permutation is also computed and synthesized separately in polynomial time. This procedure has the crucial property that the set of coefficients computed have T -count at most the T -count of the original circuit – often much lower due to coefficients in the phase polynomial adding and cancelling – hence the resulting circuit has equal or lesser T -count. In particular, we have the following lemma, which relates the T -count of a $\{\text{CNOT}, T\}$ circuit to the T -count of an implementation of the associated phase operator.

Lemma 1. *Let U_P be a linear phase operator in $\mathcal{P}_4(n)$. There exists a circuit over $\{\text{CNOT}, T\}$ implementing U_P with T -count k if and only if there exists $\mathbf{a} \in \mathbb{Z}_8^{2^n - 1}$ such that $P(\mathbf{x}) = P_{\mathbf{a}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{F}_2^n$, and \mathbf{a} has T -count at most k .*

IV. DECODING-BASED T -COUNT OPTIMIZATION

While effective at reducing T -count, it was noted that the procedure in [1] does not always find the minimal T -count, as the phase polynomial P in question may have many different representations as a weighted sum of linear Boolean functions. For instance,

$$4 \cdot x_1 + 4 \cdot x_2 + 4 \cdot (x_1 \oplus x_2) = 0 \pmod{8}$$

for all values of $x_1, x_2 \in \mathbb{F}_2$, so $P(x_1, x_2) = 4 \cdot x_1 + 4 \cdot x_2 + 4 \cdot (x_1 \oplus x_2)$ is an alternative presentation of the zero-everywhere ($\pi/4$) phase polynomial. This opens up the possibility of further T -count optimization by first finding an implementation of the target phase polynomial with a minimal number of odd coefficients, then synthesizing a circuit. By Lemma 1, this in fact finds a minimal T -count circuit. In this section we reduce this problem to a minimum-distance decoding problem and give a T -count optimization algorithm based on this decoding.

Given a tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$, we define $[\mathbf{a}]$ to be the equivalence class of implementations of $P_{\mathbf{a}}$ – i.e., $\mathbf{b} \in [\mathbf{a}]$ if and only if $P_{\mathbf{a}}(\mathbf{x}) = P_{\mathbf{b}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{F}_2^n$ (hence $U_{P_{\mathbf{a}}} = U_{P_{\mathbf{b}}}$). Moreover, we define \mathcal{C}_n to be the set of implementations of the zero-everywhere phase polynomial. Clearly for any $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$, $[\mathbf{a}] = \mathbf{a} + \mathcal{C}_n$, since by we have $P_{\mathbf{a}}(\mathbf{x}) + P_{\mathbf{b}}(\mathbf{x}) = P_{\mathbf{a}+\mathbf{b}}(\mathbf{x})$ for any $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_8^{2^n-1}$, $\mathbf{x} \in \mathbb{F}_2^n$. As a result we see that the problem of finding an implementation of $P_{\mathbf{a}}$ minimizing T count is equivalent to finding an element $\mathbf{c} \in \mathcal{C}_n$ minimizing the number of odd coefficients in $\mathbf{a} + \mathbf{c}$.

While this optimization could be performed directly over \mathcal{C}_n , we can do better by reducing the optimization problem to a decoding problem over a *binary* code where minimum-distance decoding corresponds exactly to T -count optimization. In particular, recall that the T -count of an implementation is given by the number of odd coefficients. Defining $\text{Res}_2 : \mathbb{Z} \rightarrow \mathbb{F}_2$ as the function taking the binary residue of an integer and extending this component-wise to tuples, we see that the T -count of a tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ is equal to the weight of the binary residue vector, i.e. $\text{wt}(\text{Res}_2(\mathbf{a}))$.

Since Res_2 is a ring homomorphism, we further see that

$$\text{wt}(\text{Res}_2(\mathbf{a} + \mathbf{c})) = \text{wt}(\text{Res}_2(\mathbf{a}) \oplus \text{Res}_2(\mathbf{c})) = d(\text{Res}_2(\mathbf{a}), \text{Res}_2(\mathbf{c})),$$

that, is the T -count of a tuple $\mathbf{a} + \mathbf{c}$ is the Hamming distance from $\text{Res}_2(\mathbf{a})$ to $\text{Res}_2(\mathbf{c})$. As a result, optimizing $\text{wt}(\text{Res}_2(\mathbf{a} + \mathbf{c}))$ over all $\mathbf{c} \in \mathcal{C}_n$ is exactly the problem of minimum distance decoding $\text{Res}_2(\mathbf{a})$ over $\text{Res}_2(\mathcal{C}_n)$, the set of binary residue vectors of \mathcal{C}_n . Moreover, we note that $\text{Res}_2(\mathcal{C}_n)$ is a binary linear code, since $\text{Res}_2(\mathbf{a}) \oplus \text{Res}_2(\mathbf{b}) = \text{Res}_2(\mathbf{a} + \mathbf{b}) \in \text{Res}_2(\mathcal{C}_n)$ for any $\mathbf{a}, \mathbf{b} \in \mathcal{C}_n$. In particular, this code turns out to be the $(n-4)$ th order, length $2^n - 1$ punctured Reed-Muller code:

Theorem 2. $\text{Res}_2(\mathcal{C}_n) = \mathcal{RM}(n-4, n)^*$

We defer the proof of Theorem 2 until Section V and discuss some consequences of this equivalence in the following.

A. Upper bounds

As a result of Lemma 1 and Theorem 2, the covering radius of $\text{Res}_2(\mathcal{C}_n) = \mathcal{RM}(n-4, n)^*$ gives a tight upper bound on the number of T gates required to implement a linear phase operator over $\{\text{CNOT}, T\}$. Here we mean tight in the sense that there exists a linear phase operator which requires a minimum of $\rho(\mathcal{RM}(n-4, n)^*)$ T gates to implement over $\{\text{CNOT}, T\}$. While no analytic formula has yet been found for the covering radius of higher-order Reed-Muller codes, some asymptotic upper bounds are known. The best upper bound, to the authors' knowledge, is due to Cohen and Litsyn [22]. In particular, Cohen and Litsyn showed that for large n and orders r where $n - r \geq 3$,

$$\rho(\mathcal{RM}(r, n)) \leq \frac{n^{n-r-2}}{(n-r-2)!}.$$

Since the covering radius of $\mathcal{RM}(r, n)^*$ is trivially bounded above by $\rho(\mathcal{RM}(r, n))$, we see that for sufficiently large n , $\rho(\mathcal{RM}(n-4, n)^*) \leq \frac{n^2}{2} - 1$. As a result we obtain a new asymptotic bound on the number of T gates required to implement a circuit over $\{\text{CNOT}, T\}$.

Theorem 3. *Any linear phase operator $U_p \in \mathcal{P}_4(n)$ can be implemented with $O(n^2)$ T -gates.*

B. T -count optimization

Completing a T -count optimization procedure based on Reed-Muller decoding is not as immediate. In particular, decoding the binary residue $\text{Res}_2(\mathbf{a})$ of a target tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ over $\mathcal{RM}(n-4, n)^*$ produces a minimal *residue* $\mathbf{c}' = \text{Res}_2(\mathbf{c})$ of a codeword \mathbf{c} in \mathcal{C}_n . To actually produce a minimal T -count implementation of $P_{\mathbf{a}}$ we need to compute $\mathbf{c} \in \mathcal{C}_n$ from $\text{Res}_2(\mathbf{c})$ and then synthesize $\mathbf{a} + \mathbf{c}$. Fortunately, there is an easy way to do this, given the following lemma.

Lemma 2. *For all $\mathbf{y} \in \mathbb{F}_2^n$ with $\text{wt}(\mathbf{y}) \leq n-4$ we have $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \in \mathcal{C}_n$.*

Using Lemma 2, which we prove in Section V, and the fact that the set of all monomial evaluation vectors with degree at most $n-4$,

$$\mathcal{B} = \{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n, \text{wt}(\mathbf{y}) \leq n-4\},$$

forms a basis for $\mathcal{RM}(n-4, n)^*$, we can write a decoded word \mathbf{c}' over this basis then *reinterpret* the sum over \mathbb{Z}_8 . In particular, if $\mathbf{c}' = \mathbf{b}_1 \oplus \mathbf{b}_2 \oplus \cdots \oplus \mathbf{b}_k$ for some $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k \in \mathcal{B}$, then $\mathbf{c} = \mathbf{b}_1 + \mathbf{b}_2 + \cdots + \mathbf{b}_k \in \mathcal{C}_n$, and

$$\text{Res}_2(\mathbf{c}) = \text{Res}_2(\mathbf{b}_1 + \mathbf{b}_2 + \cdots + \mathbf{b}_k) = \text{Res}_2(\mathbf{b}_1) \oplus \text{Res}_2(\mathbf{b}_2) \oplus \cdots \oplus \text{Res}_2(\mathbf{b}_k) = \mathbf{c}'.$$

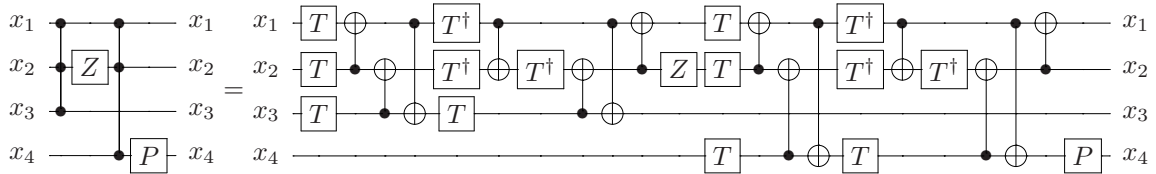
Algorithm 1 T -optimize(C)

- 1: Compute coefficients $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ from C
 - 2: $\mathbf{c}' \leftarrow \text{RM-DECODE}(n-4, n, \text{Res}_2(\mathbf{a}))$
 - 3: Write \mathbf{c}' over basis \mathcal{B} : $\mathbf{c}' = \mathbf{b}_1 \oplus \mathbf{b}_2 \oplus \cdots \oplus \mathbf{b}_k$
 - 4: $\mathbf{c} \leftarrow \mathbf{b}_1 + \mathbf{b}_2 + \cdots + \mathbf{b}_k \pmod{8}$
 - 5: $\text{SYNTHESIZE}(\mathbf{a} + \mathbf{c})$
-

Algorithm 1 summarizes our algorithm for T -count optimization in $\{\text{CNOT}, T\}$ circuits. For simplicity the algorithm assumes the input circuit implements a linear phase operator – for more general $\{\text{CNOT}, T\}$ circuits the extra linear permutation is synthesized and appended to the end. The algorithm works by computing a set of coefficients implementing the linear phase operator U_P computed by the circuit. The vector of residues modulo 2 is then decoded as \mathbf{c}' in $\mathcal{RM}(n-4, n)^*$ using the procedure $\text{RM-DECODE}(n-4, n, \text{Res}_2(\mathbf{a}))$. Any variable order Reed-Muller decoder may then be used in RM-DECODE . In particular, using a minimum distance decoder gives a minimal T -count synthesis procedure.

A vector $\mathbf{c} \in \mathcal{C}_n$ with binary residue equal to \mathbf{c}' is then computed and added to the original set of coefficients, and a circuit is synthesized for the new implementation of P . In particular, the procedure SYNTHESIZE takes a set of coefficients \mathbf{a} and synthesizes a circuit over $\{\text{CNOT}, T\}$ implementing $U_{P_{\mathbf{a}}}$. The T -par algorithm [1], for instance, may be used to implement SYNTHESIZE .

Example 2. Consider the circuit below:



By iterating through the circuit and tracking the qubit states (see, e.g., [1]), we compute the phase polynomial for this operator as

$$P(\mathbf{x}) = 2x_1 + 6x_2 + 6(x_1 \oplus x_2) + x_3 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3) \\ + 3x_4 + 7(x_1 \oplus x_4) + 7(x_2 \oplus x_4) + (x_1 \oplus x_2 \oplus x_4).$$

Writing the coefficients of P as a $2^n - 1$ -tuple \mathbf{a} we get

$$\mathbf{a} = (2, 6, 6, 1, 7, 7, 1, 3, 7, 7, 1, 0, 0, 0, 0),$$

which has a canonical T -count of 8 – a reduction of 6 T gates.

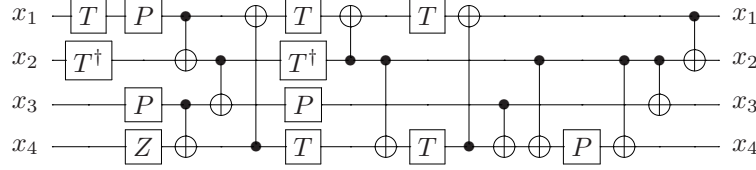
Now we optimize the implementation of P further by decoding

$$\text{Res}_2(\mathbf{a}) = (0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0)$$

in the code $\mathcal{RM}(0, 4)^*$. As $\mathcal{RM}(0, 4)^*$ is the set of evaluation vectors for degree 0 binary polynomials, there are exactly two vectors to choose from, corresponding to the zero (zero-everywhere) and constant (one-everywhere) functions. Since the all 1 vector achieves the minimum distance of 7 from $\text{Res}_2(\mathbf{a})$, we choose \mathbf{c}' to be the all 1 vector. By Lemma 2, $\mathbf{c}' = \mathbf{1} \pmod{2}$ is already in the space of zero-everywhere polynomials \mathcal{C}_n , so steps 3 & 4 are trivial and we set $\mathbf{c} = \mathbf{1} \pmod{8}$. Finally we synthesize a circuit for the tuple $\mathbf{a} + \mathbf{c} = (3, 7, 7, 2, 0, 0, 2, 4, 0, 0, 2, 1, 1, 1, 1)$, corresponding to the phase polynomial

$$P'(\mathbf{x}) = 3x_1 + 7x_2 + 7(x_1 \oplus x_2) + 2x_3 + 2(x_1 \oplus x_2 \oplus x_3) + 4x_4 + 2(x_1 \oplus x_2 \oplus x_4) \\ + (x_3 \oplus x_4) + (x_1 \oplus x_3 \oplus x_4) + (x_2 \oplus x_3 \oplus x_4) + (x_1 \oplus x_2 \oplus x_3 \oplus x_4).$$

A possible circuit implementing P' is shown below:



Note that this decoding reduces the T -count from 14 (or 8, as T -par would obtain) to 7. Moreover, the number of T gates is equal to the distance from $\text{Res}_2(\mathbf{a})$ to the decoded word \mathbf{c}' .

C. Complexity

It may be noted that Algorithm 1 gives a polynomial-time (in 2^n) reduction from T -count optimization over $\{\text{CNOT}, T\}$ to minimum-distance decoding in $\mathcal{RM}(n-4, n)^*$. We may likewise reduce the minimum-distance decoding problem for $\mathcal{RM}(n-4, n)^*$ to T -count optimization: given a binary vector $\mathbf{x} \in \mathbb{F}_2^{2^n-1}$, synthesize $U_{P_{\mathbf{x}}}$ over $\{\text{CNOT}, T\}$ then optimize the circuit and compute the coefficients $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ for the optimized circuit. As a consequence of Theorem 2, the vector $\mathbf{x} \oplus \text{Res}_2(\mathbf{a})$ is a minimum distance decoding of \mathbf{x} . Assuming the optimized circuit does not have exponentially more gates than a canonical circuit,¹ this reduction is also polynomial in the word length 2^n , so we see that the problems are in this sense *equivalent*.

This equivalence lends evidence to the difficulty of T -count optimization, even in the restricted setting of circuits over CNOT and T gates. In particular, any sub-exponential algorithm for exact optimization of T -count over n -qubit $\{\text{CNOT}, T\}$ circuits induces a polynomial-time minimum-distance decoding algorithm for $\mathcal{RM}(n-4, n)$. This can be further reduced to a *linear-time* algorithm by noting that the unitary $U_{P_{\mathbf{x}}}$

¹ The canonical circuit for any linear phase operator uses $O(n2^{n-1})$ gates.

above can be implemented with $O(2^n)$ gates using one ancilla and the Gray code to cycle through each of the 2^n binary sums of n variables with one CNOT gate each.

In either case it appears very unlikely that an efficient algorithm for minimum-distance decoding the order $n - 4$ punctured Reed-Muller code exists. No minimum distance decoding algorithms in time polynomial in 2^n or the Hamming weight of the received word are currently known for arbitrary order length 2^n binary Reed-Muller codes. While some particular orders of Reed-Muller codes have efficient decoders, e.g., order 1, it was shown in [23] that minimum-distance decoding for $\mathcal{RM}(n - 4, n)^*$ is equivalent to the problem of finding a minimal decomposition of a symmetric 3-tensor into symmetric tryads (rank 1 3-tensors), a known hard problem.

V. PROOF OF THEOREM 2

In this section we prove Theorem 2, and as a corollary Lemma 2.

A. The monomial basis

Our proof relies on a connection between the binary evaluations of polynomials over \mathbb{Z}_8 and the module $\mathbb{Z}_8^{2^n-1}$. In particular, consider the set of degree at most $n - 1$ monomial (Boolean) evaluation vectors

$$\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}\}.$$

We show that this set of vectors, under the natural inclusion of \mathbb{F}_2 in \mathbb{Z}_8 , forms a generating set for $\mathbb{Z}_8^{2^n-1}$ – moreover, since each such vector is linearly independent over $\mathbb{F}_2^{2^n-1}$ and hence also linearly independent over $\mathbb{Z}_8^{2^n-1}$, this set is in fact a basis. We call this basis the *monomial basis* and give a proof.

Lemma 3. $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}\}$ is a basis of $\mathbb{Z}_8^{2^n-1}$

Proof. We first note that the set of all non-constant monomial evaluation vectors, $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}\}$, is a basis for the module $\mathbb{Z}_8^{2^n-1}$. In particular, for any $\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}$ the vector $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ contains a leading 1 at the \mathbf{y} th index (e.g., Table I), and hence any tuple of $\mathbb{Z}_8^{2^n-1}$ may be written as a linear combination over this set. It therefore suffices to prove that $\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n$ is in the span of $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}\}$.

It may be observed that over \mathbb{F}_2 , the set of *all* monomial evaluation vectors is linearly dependent, and in particular that

$$\bigoplus_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} = \mathbf{0}$$

since every input evaluates to 1 for an even number of monomials. Further, as Res_2 is homomorphic we have

$$\bigoplus_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} = \text{Res}_2 \left(\sum_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \right) = \mathbf{0}$$

and so $\sum_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} = \mathbf{a}$ for some $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ such that $\text{Res}_2(\mathbf{a}) = 0$. If we write \mathbf{a} over the basis $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}\}$ and move all instances of $\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n$ to the left we see

$$b \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n = \mathbf{a}' - \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$$

where \mathbf{a}' is in the span of $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}, \mathbf{1}\}\}$ and $b \in \mathbb{Z}_8$.

Now suppose b is even. Then

$$\begin{aligned} (b-1) \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n &= \mathbf{a}' - \sum_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \\ \text{Res}_2(\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n) &= \text{Res}_2(\mathbf{a}') + \text{Res}_2\left(\sum_{\mathbf{y} \in \mathbb{F}_2^n} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}\right) \\ \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n &= \text{Res}_2(\mathbf{a}'). \end{aligned}$$

Since \mathbf{a}' is in the \mathbb{Z}_8 -span of $\{\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \mid \mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}, \mathbf{1}\}\}$ (i.e. the span using addition in \mathbb{Z}_8), $\text{Res}_2(\mathbf{a}')$ is in its \mathbb{F}_2 -span and may be written over this basis. However, the set of all monomial evaluation vectors of degree at least 1 is linearly independent over \mathbb{F}_2 , so we arrive at a contradiction. Thus b is odd and as such has a multiplicative inverse in \mathbb{Z}_8 and hence

$$\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n = b \cdot b^{-1} \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n = b^{-1} \cdot \left(\mathbf{a}' - \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \right).$$

□

Lemma 3 tells us that any tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ is the evaluation vector of some pseudo-Boolean polynomial function $f : \mathbb{F}_2^n \rightarrow \mathbb{Z}_8$ where $f(v_1, v_2, \dots, v_n) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n}$, and hence $\mathbf{f} = \mathbf{a} = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ in the monomial basis. Moreover, since

$$\text{Res}_2(\mathbf{a}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} \text{Res}_2(b_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}),$$

$\text{Res}_2(\mathbf{a})$ is the evaluation vector of a Boolean polynomial function with degree at most $\deg(f)$.

B. Evaluating $P_{\mathbf{a}}$

The next step in our proof is to give an analytic formula for the value of a phase function $P_{\mathbf{a}}$ applied to a vector $\mathbf{x} \in \mathbb{F}_2^n$ as a function of the degree of the polynomial form of \mathbf{a} . Specifically, we show that $P_{\mathbf{a}}(\mathbf{x})$ is equal to a linear combination of the Hamming weights – numbers of solutions – of certain Boolean polynomials arising from the multiplication of a monomial with a degree 1 polynomial.

Consider the value of a phase polynomial $P_{\mathbf{a}}$ at $\mathbf{x} \in \mathbb{F}_2^n$:

$$P_{\mathbf{a}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} (y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n).$$

We can view the above formula as an inner product, since the value $y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n = x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n$ is the \mathbf{y} th component of the evaluation vector $x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n$.

Formally, we define $\langle \mathbf{a}, \mathbf{b} \rangle$ for $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_8^{2^n-1}$ as $\sum_{i=1}^{2^n-1} a_i b_i$. Note that

$$\langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = \langle \mathbf{a}, \mathbf{c} \rangle + \langle \mathbf{b}, \mathbf{c} \rangle$$

for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{Z}_8^{2^n-1}$ since the inner product is linear in either argument over \mathbb{Z} , and hence also \mathbb{Z}_8 . Using this observation, we give an explicit formula for $P_{\mathbf{a}}(\mathbf{x})$ as a function of the monomial basis vectors appearing in \mathbf{a} :

Lemma 4. *Let $\mathbf{a} \in \mathbb{Z}_8^{2^n}$ and suppose $\mathbf{a} = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ in the monomial basis. Then*

$$P_{\mathbf{a}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)).$$

Proof.

$$\begin{aligned}
P_{\mathbf{a}}(\mathbf{x}) &= \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} (y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n) \\
&= \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} (x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)_{\mathbf{y}} \\
&= \langle \mathbf{a}, x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n \rangle \\
&= \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \langle \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}, x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n \rangle \\
&= \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)).
\end{aligned}$$

□

The value of $\text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n))$ in Lemma 4 above may be restated as the number of solutions to the equation $(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) = 1$. Fortunately, this number of a simple function of the degree of the polynomial, as the following Lemma shows.

Lemma 5. *For any $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$,*

$$\text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = 2^{n - \deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n))}$$

if $(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) \neq 0$, or 0 otherwise.

Proof. Clearly if $(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) = 0$, then $\text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = 0$ as required, so suppose instead that $(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) \neq 0$. Since $x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n$ has a degree of 1,

$$\deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}), \text{ or } \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}) + 1.$$

Consider the former case. Clearly $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ is a linear combination involving only variables present in $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$. Then by the equivalence $v_i^2 = v_i$ for any i ,

$$(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) = (\text{wt}(\mathbf{x}) \pmod{2}) \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}.$$

Since $(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) \neq 0$, it must be the case that $\text{wt}(\mathbf{x}) = 1 \pmod{2}$. As $v_{i_1} v_{i_2} \cdots v_{i_j} = 1$ exactly when $v_{i_1} = v_{i_2} = \cdots = v_{i_j} = 1$, we see that $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} = 1$ has $2^{n - \text{wt}(\mathbf{y})}$ solutions, and hence

$$\text{wt}(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}) = 2^{n - \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})}.$$

Now consider the latter case, $\deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}) + 1$. We know the linear combination $x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n$ must contain some variable not in the monomial $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$. Without loss of generality assume $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ and $x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n$ have no variables in common, as otherwise we may write

$$(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n) = (\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(c \oplus x'_1 \mathbf{v}_1 \oplus x'_2 \mathbf{v}_2 \oplus \cdots \oplus x'_n \mathbf{v}_n)$$

for some $c \in \mathbb{F}_2$ such that $x'_1 \mathbf{v}_1 \oplus x'_2 \mathbf{v}_2 \oplus \cdots \oplus x'_n \mathbf{v}_n$ involves none of the variables in $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$. Since $c \oplus x'_1 \mathbf{v}_1 \oplus x'_2 \mathbf{v}_2 \oplus \cdots \oplus x'_n \mathbf{v}_n = 1$ for exactly half of the $2^{n - \text{wt}(\mathbf{y})}$ valuations where $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} = 1$, we see that there are $2^{n - \text{wt}(\mathbf{y}) - 1}$ solutions, hence

$$\text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = 2^{n - \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}) - 1}$$

as required. □

In general, it is not the case that $\text{wt}(\mathbf{f}) = 2^{n-\deg(f)}$ for an n -variate Boolean polynomial function f . In particular, consider $f(v_1, v_2, \dots, v_n) = 1 \oplus v_1 v_2 \cdots v_i$. Since $v_1 v_2 \cdots v_i = 1$ has 2^{n-i} solutions,

$$\text{wt}(\mathbf{f}) = 2^n - 2^{n-i} \neq 2^{n-\deg(f)}.$$

As a corollary to Lemmas 4 and 5 we obtain a proof of Lemma 2.

Corollary 1 (Lemma 2). *For all $\mathbf{y} \in \mathbb{F}_2^n$ with $\text{wt}(\mathbf{y}) \leq n-4$ we have $\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n} \in \mathcal{C}_n$.*

Proof. Let $\mathbf{a} = \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ for some $\mathbf{y} \in \mathbb{F}_2^n$ of Hamming weight at most $n-4$ and let $\mathbf{x} \in \mathbb{F}_2^n$. By Lemma 4,

$$P_{\mathbf{a}}(\mathbf{x}) = \text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n))$$

Since $\text{wt}((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = 0$ or $2^{n-\deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n))}$ by Lemma 5 and $\deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) \leq n-3$ we have $P_{\mathbf{a}}(\mathbf{x}) = 0 \pmod{8}$ as required. \square

C. Proof of Theorem 2

From Lemma 5 it's immediate that if a tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ may be written over the monomial basis with degree at most $n-4$, then $P_{\mathbf{a}}(\mathbf{x}) = 0 \pmod{8}$ for any \mathbf{x} and so $\mathbf{a} \in \mathcal{C}_n$. However, it may be the case that \mathbf{a} contains monomials with degree greater than $n-4$ and yet are still in the codespace \mathcal{C}_n . For instance, consider $\mathbf{a} = 2 \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3}$. For any $\mathbf{x} \in \mathbb{F}_2^n$,

$$\begin{aligned} P_{\mathbf{a}}(\mathbf{x}) &= 2 \cdot \text{wt}((\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) \\ &= 2 \cdot 2^{n-\deg((\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n))} \\ &= 0 \pmod{8}. \end{aligned}$$

In this case we have $\mathbf{a} \in \mathcal{C}_n$ and $\text{Res}_2(\mathbf{a}) = \mathbf{0} \in \mathcal{RM}(n-4, n)^*$ for $n \geq 4$ even though as a polynomial over \mathbb{Z}_8 , \mathbf{a} has degree greater than $n-4$. On the other hand, in some sense, with regard to Lemma 5, the term $2 \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3} = 2^1 \cdot \mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3}$ has an *effective* degree of $\deg(\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3}) - 1 = n-4$, since $2 \cdot \text{wt}((\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_{n-3})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = 0$, $2^{n-(n-4)} = 2^4$ or $2^{n-(n-3)} = 2^3$.

We define the effective degree of a term of the form $2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ to be $\text{wt}(\mathbf{y}) - i$. Moreover, we let the effective degree of a polynomial sum $\sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ to be the highest effective degree of any term obtained by expanding each coefficient $b_{\mathbf{y}}$ to its binary representation,

$$b_{\mathbf{y}} = (b_{\mathbf{y}})_0 2^0 + (b_{\mathbf{y}})_1 2^1 + (b_{\mathbf{y}})_2 2^2.$$

As we prove below, the phase polynomial associated with a tuple $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ necessarily evaluates to a non-zero value mod 2^k for some input if $\mathbf{a} = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}} b_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ has effective degree $n-k$.

Lemma 6. *Let $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$ have effective degree $n-k$ in the monomial basis. There exists $\mathbf{x} \in \mathbb{F}_2^n$ such that*

$$P_{\mathbf{a}}(\mathbf{x}) \not\equiv 0 \pmod{2^k}.$$

Proof. Suppose to the contrary that $P_{\mathbf{a}}(\mathbf{x}) \equiv 0 \pmod{2^k}$ for all $\mathbf{x} \in \mathbb{F}_2^n$. Consider a term $2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ having effective degree $n-l$ for some l greater than or equal to k , hence the effective degree is at most $n-k$. By Lemmas 4 and 5,

$$P_{2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}}(\mathbf{x}) = 2^l \text{ or } 2^{l-1},$$

and since $l \geq k$, the result is non-zero mod 2^k if and only if $l = k$ (i.e., the effective degree is $n-k$) and $P_{2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}}(\mathbf{x}) = 2^{l-1}$. From Lemma 5 we know that this is the case exactly if

$$\deg((\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n})(x_1 \mathbf{v}_1 \oplus x_2 \mathbf{v}_2 \oplus \cdots \oplus x_n \mathbf{v}_n)) = \deg(\mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}) + 1,$$

or equivalently the sum $x_1v_1 \oplus x_2v_2 \oplus \cdots \oplus x_nv_n$ contains a non-zero multiple of some variable *not* present in the monomial $v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}$. It can then be observed that for any input $\mathbf{x} \in \mathbb{F}_2^n$, if $P_{\mathbf{a}}(\mathbf{x}) = 0 \pmod{2^k}$, there must be an even number of terms with effective degree $n - k$ that do not contain some variable in the sum $x_1v_1 \oplus x_2v_2 \oplus \cdots \oplus x_nv_n$. We show that this is impossible for all $\mathbf{x} \in \mathbb{F}_2^n$ by an inclusion-exclusion argument.

We define S_i to be the set of all effective degree $n - k$ terms of \mathbf{a} that do not contain the variable v_i . Clearly $\cup_{i|x_i=1} S_i$ gives the set of all such terms that do not contain some variable in the sum $x_1v_1 \oplus x_2v_2 \oplus \cdots \oplus x_nv_n$. Moreover, $|\cup_{i|x_i=1} S_i|$ gives the number of such terms. By the assumption and the observation above that $P_{\mathbf{a}}(\mathbf{x}) = 0 \pmod{2^k}$ if and only if there are an even number of terms in \mathbf{a} with effective degree $n - k$ that do not contain some variable in the sum $x_1v_1 \oplus x_2v_2 \oplus \cdots \oplus x_nv_n$, it follows that for any $\mathbf{x} \in \mathbb{F}_2^n$,

$$|\cup_{v_i \in A} S_i| = 0 \pmod{2}.$$

Now take some term $2^i \cdot v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}$ of effective degree $n - k$ in \mathbf{a} but with minimal *actual* degree – that is, the degree of the monomial part, $v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}$. Since $v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}$ has minimal degree, every other term of effective degree $n - k$ must contain some variable v_i for which $y_i = 0$. Hence, the intersection of S_i over all indexes for which v_i is not in the monomial $v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}$ contains exactly one term,

$$\cap_{i|y_i=0} S_i = \{2^i \cdot v_1^{y_1}v_2^{y_2} \cdots v_n^{y_n}\}.$$

Now, $|\cap_{i|y_i=0} S_i|$ can be written as a sum of cardinalities of unions of the sets S_i :

$$|\cap_{i|y_i=0} S_i| = \sum_{\mathbf{x} \in \mathbb{F}_2^n} s_{\mathbf{x}} |\cup_{i|x_i=1} S_i|$$

for some integers $s_{\mathbf{x}}$. For instance, $|S_1 \cap S_2| = |S_1| + |S_2| - |S_1 \cup S_2|$. However, since $|\cup_{i|x_i=1} S_i| = 0 \pmod{2}$ for any \mathbf{x} , we have $|\cap_{i|y_i=0} S_i| = 0 \pmod{2}$, a contradiction. Thus there exists $\mathbf{x} \in \mathbb{F}_2^n$ such that $P_{\mathbf{a}}(\mathbf{x}) \neq 0 \pmod{2^k}$, as required. \square

With Lemma 6 we have everything we need to prove Theorem 2 below.

Proof of Theorem 2.

$\mathcal{RM}(n - 4, n)^* \subseteq \text{Res}_2(\mathcal{C}_n)$:

Let \mathbf{c}' be in $\mathcal{RM}(n - 4, n)^*$. Then

$$\mathbf{c}' = \bigoplus_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}, \text{wt}(\mathbf{y}) \leq n-4} a_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$$

for some coefficients $a_{\mathbf{y}} \in \mathbb{F}_2$. If we let

$$\mathbf{c} = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{1}\}, \text{wt}(\mathbf{y}) \leq n-4} a_{\mathbf{y}} \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n},$$

then by Lemma 2, \mathbf{c} is in \mathcal{C}_n . Moreover, $\mathbf{c}' = \text{Res}_2(\mathbf{c})$, hence $\mathbf{c}' \in \text{Res}_2(\mathcal{C}_n)$.

$\text{Res}_2(\mathcal{C}_n) \subseteq \mathcal{RM}(n - 4, n)^*$:

Let \mathbf{a} be an element of \mathcal{C}_n . By definition of \mathcal{C}_n , $P_{\mathbf{a}}(\mathbf{x}) = 0 \pmod{2^3}$ for all $\mathbf{x} \in \mathbb{F}_2^n$. However, by Lemma 6, we know there exists $\mathbf{x} \in \mathbb{F}_2^n$ such that $P_{\mathbf{a}}(\mathbf{x}) \neq 0 \pmod{2^k}$ if \mathbf{a} has effective degree $n - k$ in the monomial basis – hence, \mathbf{a} can have effective degree at most $n - 4$.

Now, suppose \mathbf{a} may be written as some sum $\sum 2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$ of effective degree at most $n - 4$ – that is, $\text{wt}(\mathbf{y}) - i \leq n - 4$ for every pair of values \mathbf{y} and i in the summation. Then

$$\text{Res}_2(\mathbf{a}) = \bigoplus_{i=0} 2^i \cdot \mathbf{v}_1^{y_1} \mathbf{v}_2^{y_2} \cdots \mathbf{v}_n^{y_n}$$

which has degree at most $n - 4$. Hence $\text{Res}_2(\mathbf{a}) \in \mathcal{RM}(n - 4, n)^*$. \square

VI. GENERALIZATION

In this section we discuss the generalization of Theorem 2 to linear phase circuits with smaller angle Z rotations. Specifically, the gate $R_z(\pi/2^k)$ for any non-negative integer k is defined as

$$R_z(\pi/2^k) : |x\rangle \mapsto e^{i\frac{\pi}{2^k}x}|x\rangle.$$

We show that the minimal number of $R_z(\pi/2^k)$ gates required to implement a unitary over $\{\text{CNOT}, R_z(\pi/2^k)\}$ corresponds to a minimum distance decoding in $\mathcal{RM}(n-k-2, n)^*$. Such gates arise, e.g., in Shor's algorithm [7] and the Clifford hierarchy [24]. Moreover, researchers have recently developed state distillation techniques for these gates, allowing them to be performed fault tolerantly without approximating them over another gate set [25, 26].

We define $\mathcal{P}_{2^k}(n)$ to be the set of n -qubit $\pi/2^k$ linear phase operators – that is, n -qubit diagonal unitary matrices implementable over $\{\text{CNOT}, R_z(\pi/2^k)\}$. As in the case of $\pi/4$ linear phase operators, such an operator applies to each basis vector a phase rotation that is a 2^k th root of unity determined by a linear combination of linear functions of its bits. In particular, for any $U \in \mathcal{P}_{2^k}$, U has the following affect:

$$U : |\mathbf{x}\rangle \mapsto e^{i\frac{\pi}{2^k}P(\mathbf{x})}|\mathbf{x}\rangle, \quad P(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{0\}} a_{\mathbf{y}} \cdot (y_1x_1 \oplus y_2x_2 \oplus \cdots \oplus y_nx_n)$$

for some coefficients $\mathbf{a} \in \mathbb{Z}_{2^k}^{2^n-1}$. As before we call the tuple \mathbf{a} an implementation of P .

In Section III, the canonical circuit for an implementation of a $\pi/4$ linear phase operator was defined by computing $y_1x_1 \oplus y_2x_2 \oplus \cdots \oplus y_nx_n$ for each nonzero $a_{\mathbf{y}}$, then applying a sequence of T , P and Z gates to achieve the correct power of $e^{i\frac{\pi}{4}}$. We may define the canonical circuit for an implementation of any $\pi/2^k$ linear phase operator in the same way: compute $y_1x_1 \oplus y_2x_2 \oplus \cdots \oplus y_nx_n$ then apply $R_z(\pi/2^k)^l$ to achieve the correct power of $e^{i\frac{\pi}{2^k}}$. Under the reasonable assumption that $R(\pi/2^k)$ gates are more expensive than $R(\pi/2^{k'})$ gates whenever $k > k'$, we define

$$R_z(\pi/2^k)^l := R_z(\pi)^{l_k} \cdots R_z(\pi/2^{l-1})^{l_1} R_z(\pi/2^k)^{l_0}$$

where $l_k \cdots l_1 l_0$ is the binary expansion of l . Denoting by $\mathbf{a} \gg i$ the component-wise quotient of \mathbf{a} divided by 2^i , we find that the number of $R_z(\pi/2^l)$ gates in the canonical circuit is $\text{wt}(\text{Res}_2(\mathbf{a} \gg (k-l)))$ – the number of components $a_{\mathbf{y}}$ that have a 1 in the $(k-l)$ th digit of their binary expansion. In particular, the number of $R_z(\pi/2^k)$ gates is just $\text{wt}(\text{Res}_2(\mathbf{a}))$, which as in the case of $\{\text{CNOT}, T\}$ circuits may be optimized by finding a minimum distance codeword in the space of all zero-mod- 2^{k+1} polynomials,

$$\mathcal{C}_n^k = \{\mathbf{c} \in \mathbb{Z}_{2^{k+1}}^{2^n-1} \mid P_{\mathbf{c}}(\mathbf{x}) = 0 \pmod{2^{k+1}} \text{ for all } \mathbf{x} \in \mathbb{F}_2^n \setminus \{0\}\}.$$

It turns out that this residue is in fact the order $n-k-2$, length 2^n-1 punctured Reed-Muller code², $\mathcal{RM}(n-k-2, n)^*$.

Theorem 4.

$$\mathcal{RM}(n-k-2, n)^* = \text{Res}_2(\mathcal{C}_n^k)$$

We do not give a formal proof of Theorem 4, but note that the proof of Theorem 2 suffices with a few simple modifications. In particular, rather than a vector over \mathbb{Z}_8 we have a vector over $\mathbb{Z}_{2^{k+1}}$ and the phase polynomial $P_{\mathbf{a}}$ is evaluated over $\mathbb{Z}_{2^{k+1}}$.

As a natural consequence of Theorem 4, the number of rotation gates of any angle $\pi/2^l$ for $l \leq k$ may be reduced by decoding $\text{Res}_2(\mathbf{a} \gg (k-l))$ in the code $\text{Res}_2(\mathcal{C}_n^l) = \mathcal{RM}(n-l-2, n)$ and adding the decoded tuple back into \mathbf{a} (multiplied by the appropriate power of 2). Such procedures may be a valuable tool for

² Note that using Definition 2, the Reed-Muller code $\mathcal{RM}(r, n)$ is well defined for $r < 0$. In particular, the code is the trivial code $\{0\}$, corresponding to the fact that no non-trivial zero phase polynomials exist mod 2^k when $k < n-1$.

quantum circuits utilizing progressively finer grain Z rotations, such as Shor’s algorithm [7], either to be later approximated by Clifford+ T gates or to be performed directly using state distillation. One potential issue with this method is reducing the number of $R_z(\pi/2^l)$ may increase the number of $R_z(\pi/2^{l'})$ gates for any $l' < l$, as seen in Example 2. In most cases smaller angles of rotation are more costly so this is a reasonable trade off, but we leave it as an open question to find a general algorithm for optimizing the total cost of all rotation gates in a $\{\text{CNOT}, R_z(\pi/2^k)\}$ circuit.

VII. EXPERIMENTS

We implemented Algorithm 1 in T -par [5] as an optimization pass in the resynthesis procedure. T -par optimizes circuits over the Clifford+ T gate set by computing a representation using exponential sums, then resynthesizing. As our algorithm presently applies to CNOT and phase gates, we break up the input circuit into $\{\text{CNOT}, T\}$ subcircuits, each of which is then optimized individually.

We implemented and tested the algorithm with two Reed-Muller decoders – the original majority logic decoder [21] due to Reed, and a modern recursive decoder [27] due to Dumer. The former has complexity in $O(2^{2n})$ for an n -qubit circuit while the latter has a significantly lower complexity of $O(2^n)$. While both of these algorithms are exponential in the number of qubits n , we nonetheless obtain reasonable performance for large circuits by storing and operating directly on compressed vector representations. In order to optimize these large circuits we chose relatively fast decoders over minimum-distance decoders.

A. Evaluation

Algorithm 1 was evaluated on a suite of benchmark quantum circuits, drawn from the literature and the Reversible Logic Benchmarks page [28]. The majority of circuits tested are reversible circuits, though some specifically quantum circuits were also examined. Toffoli gates were replaced with a Clifford+ T implementation using 7 T -gates [4], and multiple control Toffolis were expanded into two-control Toffoli gates using one zero initialized ancilla (see, e.g., [18]).

Table II reports the T -count of circuits optimized with both T -par alone, and with Algorithm 1 using either the majority logic or recursive decoder applied to $\{\text{CNOT}, T\}$ subcircuits. All experiments were run on with a 2.4GHz quad-core Intel Core i7 processor running Linux and 8GB of RAM. Each benchmark had a timeout of 30 minutes – instances where the algorithm failed to report a result within the timeout are identified with a dash.

On average, Algorithm 1 reduced T -count by 6% for both the majority logic decoder and the recursive decoder compared to T -par. While the recursive decoder produced the best results in some cases, notably the GF multipliers, and failed less often, for many benchmarks it reported significantly *increased* T -counts compared to T -par. Majority logic decoding by comparison typically produced less T -reduction, though it consistently resulted in circuits with equal or lesser T -count than that reported by T -par. Counter-intuitively this appears to result from the recursive decoder actually doing a *better* job optimizing T -count – after the recursive decoder performs significant rewrites on individual $\{\text{CNOT}, T\}$ subcircuits, T -par has less opportunity to optimize T -gates across subcircuit boundaries. A natural direction of future research is to extend decoding-based optimization to $\{H, \text{CNOT}, T\}$ circuits in order to make use of the additional T -count reductions possible across subcircuit boundaries.

While the T -count reductions over T -par are minor compared to the initial jump from the original T -count, the results clearly demonstrate that further T -count optimization beyond the T -par algorithm is possible. In the most significant case, a T -count reduction of 75% was reported for the benchmark BCSD₈ with both decoders. Note that it may be possible to achieve better T -count with other Reed-Muller decoders as well. We leave exploration of effective Reed-Muller decoders as an avenue for future work.

³ Grover’s search is performed with 4 iterations using the oracle $f(\vec{x}) = \neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5$.

Benchmark	n	T -count (original)	T -count (T -par)	T -count (majority)	T -count (recursive)
Grover ₅ [29] ^a	9	140	52	52	52
Mod 5 ₄ [28]	5	28	16	16	16
VBE-Adder ₃ [30]	10	70	24	24	24
CSLA-MUX ₃ [31]	15	70	62	62	58
CSUM-MUX ₉ [31]	30	196	140	84	76
QCLA-Com ₇ [32]	24	203	95	94	153
QCLA-Mod ₇ [32]	26	413	249	238	299
QCLA-Adder ₁₀ [32]	36	238	162	—	188
Adders ₈ [33]	24	399	215	213	249
RC-Adder ₆ [34]	14	77	63	47	47
Mod-Red ₂₁ [35]	11	119	73	73	73
Mod-Mult ₅₅ [35]	9	49	37	35	35
Mod-Adder ₁₀₂₄ [28]	28	1995	1011	1011	1011
BCSD ₂ [36]	9	14	14	2	2
BCSD ₄ [36]	14	20	20	4	4
BCSD ₈ [36]	21	32	32	8	8
Cycle 17_3 [28]	35	4739	1945	1944	1982
GF(2 ⁴)-Mult [37]	12	112	68	68	68
GF(2 ⁵)-Mult [37]	15	175	111	111	101
GF(2 ⁶)-Mult [37]	18	252	150	150	144
GF(2 ⁷)-Mult [37]	21	343	217	217	208
GF(2 ⁸)-Mult [37]	24	448	264	264	237
GF(2 ⁹)-Mult [37]	27	567	351	—	301
GF(2 ¹⁰)-Mult [37]	30	700	410	—	410
GF(2 ¹⁶)-Mult [37]	48	1792	1040	—	—
GF(2 ³²)-Mult [37]	96	7168	4128	—	—
Hamming ₁₅ (low) [28]	17	161	97	97	97
Hamming ₁₅ (med) [28]	17	574	230	230	230
Hamming ₁₅ (high) [28]	20	2457	1019	1019	1019
HWB ₆ [28]	7	105	71	75	75
HWB ₈ [28]	12	5887	3551	3531	3531
n th-prime ₆ [28]	9	812	402	400	400
n th-prime ₈ [28]	12	6671	4047	4045	4045
QFT ₄ [18]	5	69	67	67	67
$\Lambda_3(X)$ – [38]	5	28	16	16	16
– [18]	5	21	15	15	15
$\Lambda_4(X)$ – [38]	7	56	28	28	28
– [18]	7	35	23	23	23
$\Lambda_5(X)$ – [38]	9	84	40	40	40
– [18]	9	49	31	31	31
$\Lambda_{10}(X)$ – [38]	19	224	100	100	100
– [18]	19	119	71	71	71

TABLE II. T -count optimization results. n reports the number of qubits in the circuit. T -counts are recorded for the original circuit, after optimization by T -par, and after optimization by Algorithm 1 with either the majority logic or recursive decoder.

As an additional note, while we don’t consider T -depth optimization in this paper, reductions to T -count in some benchmarks allow further reductions to T -depth using matroid partitioning. In the extreme case, T -depth in CSUM-MUX₉ was reduced from 11 to 6 using the recursive decoder, providing strong evidence that reducing T -count is an effective means of optimizing T -depth.

ACKNOWLEDGMENTS

Matthew Amy is supported in part by the Natural Sciences and Engineering Research Council of Canada. Michele Mosca is supported by Canada’s NSERC, MPrime, CIFAR, and CFI. IQC and Perimeter Institute are supported in part by the Government of Canada and the Province of Ontario.

[1] Matthew Amy, Dmitri Maslov, and Michele Mosca, “Polynomial-time t -depth optimization of Clifford+T circuits via matroid partitioning,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **33**, 1476–1489 (2013) arXiv:1303.2042.

- [2] Adam Paetznick and Ben W. Reichardt, “Universal fault-tolerant quantum computation with only transversal gates and error correction,” ArXiv e-prints (2013), arXiv:1304.3709.
- [3] Alexei Y. Kitaev, “Fault-tolerant quantum computation by anyons,” *Annals of Physics* **303**, 2 – 30 (2003).
- [4] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler, “A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* **32**, 818–830 (2013), arXiv:1206.0758.
- [5] Matthew Amy, “T-par - a quantum circuit optimizer based on sum-over-paths representations,” .
- [6] Simon Forest, David Gosset, Vadym Kliuchnikov, and David McKinnon, “Exact synthesis of single-qubit unitaries over clifford-cyclotomic gate sets,” *Journal of Mathematical Physics* **56**, 082201 (2015), arXiv:1501.04944.
- [7] Peter W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (1994) pp. 124–134, quant-ph/9508027v2.
- [8] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo, “An algorithm for the t-count,” *Quantum Info. Comput.* **14**, 1261–1276 (2014), arXiv:1308.4134.
- [9] Peter Selinger, “Quantum circuits of T -depth one,” *Phys. Rev. A* **87**, 042302 (2013), arXiv:1210.0974.
- [10] Nabila Abdessaied, Mathias Soeken, and Rolf Drechsler, “Quantum circuit optimization by hadamard gate reduction,” in *Reversible Computation, Lecture Notes in Computer Science*, Vol. 8507, edited by Shigeru Yamashita and Shin-ichi Minato (Springer International Publishing, 2014) pp. 149–162.
- [11] Dmitri Maslov, “On the advantages of using relative phase toffolis with an application to multiple control toffoli optimization,” arXiv preprint (2015), arXiv:1508.03273.
- [12] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca, “Fast and efficient exact synthesis of single-qubit unitaries generated by clifford and t gates,” *Quantum Info. Comput.* **13**, 607–630 (2013), arXiv:1206.5236.
- [13] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca, “Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits,” *Phys. Rev. Lett.* **110**, 190502 (2013), arXiv:1212.0822.
- [14] Peter Selinger, “Efficient clifford+ t approximation of single-qubit operators,” *Quantum Info. Comput.* **15**, 159–180 (2015), arXiv:1212.6253.
- [15] Neil J Ross and Peter Selinger, “Optimal ancilla-free clifford+ t approximation of z -rotations,” arXiv preprint (2014), arXiv:1403.2975.
- [16] Adam Paetznick and Krysta M. Svore, “Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries,” *Quantum Info. Comput.* **14**, 1277–1301 (2014), arXiv:1311.1074.
- [17] Alex Bocharov, Martin Roetteler, and Krysta M. Svore, “Efficient synthesis of universal repeat-until-success quantum circuits,” *Phys. Rev. Lett.* **114**, 080502 (2015), arXiv:1404.5320.
- [18] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2000).
- [19] Florence J. MacWilliams and Neil J.A. Sloane, *The Theory of Error Correcting Codes*, Mathematical Studies No. v. 2, pt. 2 (North-Holland Publishing Company, 1978).
- [20] David E. Muller, “Application of boolean algebra to switching circuit design and to error detection,” *Electronic Computers, Transactions of the I.R.E. Professional Group on* **EC-3**, 6–12 (1954).
- [21] Irving Reed, “A class of multiple-error-correcting codes and the decoding scheme,” *Information Theory, Transactions of the IRE Professional Group on* **4**, 38–49 (1954).
- [22] Gérard D. Cohen and Simon N. Litsyn, “On the covering radius of reed-muller codes,” *Discrete Mathematics* **106**, 147 – 155 (1992).
- [23] Gadiel Seroussi and Abraham Lempel, “Maximum likelihood decoding of certain reed - muller codes,” *Information Theory, IEEE Transactions on* **29**, 448–450 (1983).
- [24] Daniel Gottesman and Isaac L. Chuang, “Quantum teleportation is a universal computational primitive,” *Nature* , 390–393 (1999), quant-ph/9908010.
- [25] Andrew J. Landahl and Chris Cesare, “Complex instruction set computing architecture for performing accurate quantum z rotations with less magic,” arXiv preprint (2013), arXiv:1302.3240.
- [26] Guillaume Duclos-Cianci and David Poulin, “Reducing the quantum-computing overhead with complex gate distillation,” *Phys. Rev. A* **91**, 042315 (2015).
- [27] Ilya Dumer, “Recursive decoding and its performance for low-rate reed-muller codes,” *Information Theory, IEEE Transactions on* **50**, 811–823 (2004).
- [28] Dmitri Maslov, “Reversible logic synthesis benchmarks page,” (2011).
- [29] Lov K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC ’96 (ACM, New York, NY, USA, 1996) pp. 212–219.

- [30] Vlatko Vedral, Adriano Barenco, and Artur Ekert, “Quantum networks for elementary arithmetic operations,” *Phys. Rev. A* **54**, 147–153 (1996), quant-ph/9511018.
- [31] Rodney Van Meter and Kohei M. Itoh, “Fast quantum modular exponentiation,” *Phys. Rev. A* **71**, 052320 (2005), quant-ph/0408006.
- [32] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore, “A logarithmic-depth quantum carry-lookahead adder,” *Quantum Info. Comput.* **6**, 351–369 (2006), quant-ph/0406142.
- [33] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro, “Quantum addition circuits and unbounded fan-out,” *Quantum Info. Comput.* **10**, 872–890 (2010), arXiv:0910.2530.
- [34] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton, “A new quantum ripple-carry addition circuit,” *ArXiv e-prints* (2004), quant-ph/0410184.
- [35] Igor L. Markov and Mehdi Saeedi, “Constant-optimized quantum circuits for modular multiplication and exponentiation,” *Quantum Info. Comput.* **12**, 361–394 (2012), arXiv:1202.6614.
- [36] Austin G Fowler, Simon J Devitt, and Cody Jones, “Surface code implementation of block code state distillation,” *Scientific reports* **3** (2013), arXiv:1301.7107.
- [37] Dmitri Maslov, Jimson Mathew, Donny Cheung, and Dhiraj K. Pradhan, “An $O(m^2)$ -depth quantum algorithm for the elliptic curve discrete logarithm problem over $\text{GF}(2^m)$,” *Quantum Info. Comput.* **9**, 610–621 (2009), arXiv:0710.1093.
- [38] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter, “Elementary gates for quantum computation,” *Phys. Rev. A* **52**, 3457–3467 (1995), quant-ph/9503016.